Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission to download and/or print my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

The robotic hand parts database Design

by

Xiangwei He

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in the Department of Engineering

Idaho State University

Fall 2015

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Xiangwei He find it satisfactory and recommend that it be accepted.

_____

_ Alba Perez Gracia

Major Advisor

_____

_ Anish Sebastian

Committee Member

_____

_ Robert Fisher, Jr.

Graduate Faculty Representative

Content

List of Figures

Tables List

Abstract

The purpose of this project is to build a database for the automatic design of robotic hands in a solid modeling environment. This is a simple database that includes fingertips, links, palms, bases, and pins. It is aimed at helping people who are interested in robotic design. This database gives them the chance to build a robotic model without proficient knowledge background in detailed design. The parts are created as a preliminary design that allows assessing the motion and performance of the hand. User also can edit the parts and add new part in the database. This database can work independently or interacting with a hand synthesis program to help the user in the building of hand models.

Chapter 1

Introduction

As we all know, one of the main differences between human and other animals is that human can create and use tools. This not only because we have a complicated brain, but also because we have a pair of dexterous hands. Scientists are trying to generate an artificial intelligence machine to mimic the human's brain, and similar efforts are devoted to build a robotic hand which can work with the versatility of the human hand.

After the Second Industrial Revolution, world progress with society electrification. And the combination of electronic technology and mechanical technology gives people the chance to make the robotic dream come true. The first autonomous robots which had a simple function were created by William Grey Walter in 1948 and 1949. After that, with the development of the computer science and electronic components like sensors, processors, and small actuators, people finally can build robotic hands similar to the human hand.

In these years, rapid proto-typing technology is developing very fast, especially 3D-printers. As they become more and more common in our life, people can freely build models that they like. And if more people can have interest in making or studying robotic hands, it will make the field of robotic hands increase in their diversity and applications. But for normal people, not all of them know how to use modeling software, and sometimes they may not have enough time

to be skilled about a software. The aim of this project is to build a hand parts database to help people build the robotic hands at their will.

For most researchers who are working in robotic hands, the aim is to build special hands. That means they build them for special requirements. Like the Scott Hand [1], which was designed and made by Kurt Scott. He made this hand based on the real data of human hands, and

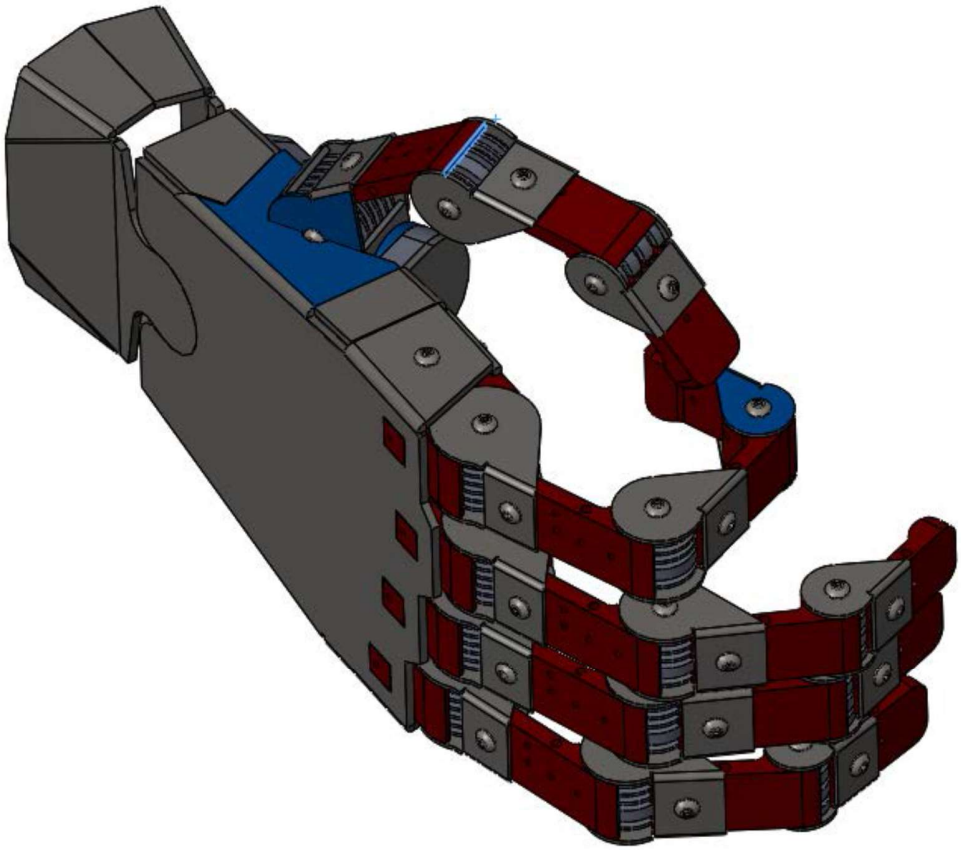


Figure 1.The model of Scott Hand

it can simulate several motions of our hands. This is a well-finished model in which the detailed design was a big component. Comparing with this kind of unchangeable model, the aim of this research is that the user could build a personal model base on the database. People can define different dimensions for all parts, and people also can do some modification after the parts are generated. People can use this database to build a humanoid hand, like the Scott hand, or also can build other type of reduced-mobility hands such as the BarrettHand[2].

Figure 2. The BarrettHand[2]

Chapter 2

Robotic hand kinematics

From a kinematics point of view, a robotic hand is a kind of mechanical grip, it is programmable and can finish several tasks, include motion under automatic control. Since the robotic hand is a kind of robot end-effector, we can use robot kinematics to analyze it. The main characteristic of a robot is its capability of movement in a six-dimensional space that includes translational and rotational coordinates.

No matter what parts are to be included in a robotic hand, they are all connected by joints. That means that all hand parts are composed by two things: rigid body and joint. Rigid body section characterized the shape and working surface of a hand part, and the joint restricts the relative movement of two connected hand parts.

2.1 Degrees of freedom and Workspace

The degrees of freedom of the robot, also called mobility, are defined as the number of independent parameters needed to specify the positions of all members of the system relative to a base frame. The most commonly used criteria to find the mobility of mechanisms is the Kutzbach-Gruebler formula. For a robot with *n* links (counting the base) and *j* joints, in which each joint *i* allows f_i degrees of freedom, we compute the degrees of freedom as

$$\boldsymbol{M = 6(n-1) - \sum_{i=1}^{j}(6 - f_i).} \qquad \textbf{(1)}$$

For a serial robot, the degrees of freedom are equal to the number of joints multiplied by the mobility allowed by each joint. The mobility indicates how many of the joints of the robot need to be actuated, also called active joints. In a serial robot, all joints are active.

After mobility and active joints are calculated, the next part is to analyze the direct kinematics and the inverse kinematics. In the direct kinematics, which is also called forward kinematics, we will use a matrix to relate the position of end-effector to the rotation or translation of each joint. The end-effector can be totally defined by the joint variables. In the inverse kinematics, the key is calculating the joint variables to reach a given point.

The workspace is defined by the result of the direct kinematics. Once we have all dimensions of the joints, like position, type and range, we can denote a region which is a combination of all possible positions of end-effector. It is important that position includes location and orientation; the workspace of the robot is a six-dimensional subset of the six-dimensional space of rotations and translations.

Because of the difficulty in visualizing the workspace, several subspaces have been defined [].

- Reachable workspace (W_R): Set of all locations of the origin of the end-effector frame that the robot can reach. It is a three-dimensional subset of the workspace.
- Dexterous workspace (W_D): Set of all locations of the origin of the end-effector frame that the robot can reach with any orientation. It is useful because the robot has full dexterity in this subspace, which ensures that any task can be performed within it.

- Workspace with constant orientation (W_θ): Set of all locations of the origin of the end-effector frame that the robot can reach with a specified orientation for the end-effector.

And we have

$$\cup_{\theta \in range} \boldsymbol{W_\theta} = \boldsymbol{W_R}\ ,\ \cap_{\theta \in range} \boldsymbol{W_\theta} = \boldsymbol{W_D} \qquad (2)[3]$$

2.2 Basic Joint

For easily understanding the joints, a list is included with the six basic types of joints that continuously share a surface of contact, which were identified by Releaux[4]. He also called them lower pairs and his classification is still used today in most robotics and mechanisms books. He defined revolute (R), cylindrical (C), prismatic (P), spherical (S), helical (H) and plane (E) joints. In our research, we only consider the basic types of revolute and prismatic joints; any of the others can be obtained by combining the two basic types with certain additional constraints. In addition to these, we refer to the universal joint (T), which can be modeled as two revolute joints at right angles. See Figure 3.[3]

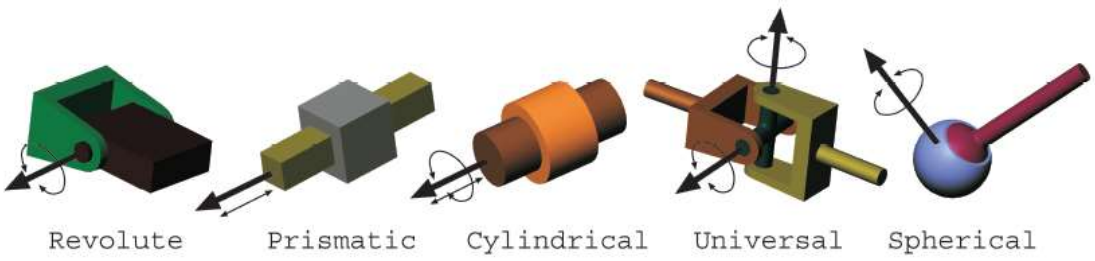


Figure 3. Common type of joints

The revolute joint is a one-degree-of-freedom joint, which allows a rotation of angle θ around an arbitrary located joint axis. The prismatic joint is also a one-degree-of-freedom joint that allows movement along the direction of a line. It is interesting that the translation of the prismatic joint is not affect by space position of prismatic axis. The cylindrical joint is a general screw motion, in which the rotation around and the translation along the screw axis *S* are nonzero and independent. It can be constructed as the composition of a revolute joint and a prismatic joint with same direction. In a similar fashion, the universal (T) joint and the spherical (S) joint can be constructed from individual rotations.

2.3 Line Geometry and the Screw Axis

Form last section, we can know that several types of joints have an invariant line, known as the joint axis. At this point, we are interested in having a good, useful definition of an axis in space. So we use a description of a line which is called the *Plücker coordinates*. They use two vectors to define a line: a direction vector and a moment vector,

$$\mathbf{L} = \mathbf{s} + \varepsilon \boldsymbol{s}^{\mathbf{0}} = \boldsymbol{s} + \varepsilon \mathbf{c} \times \mathbf{s}, \qquad \textbf{(3)[3]}$$

Where $\mathbf{s}$ is a three-dimensional vector performing the direction of the line and $\mathbf{s^0}$ is also a three-dimensional vector, which is indicating the moment of the line. The moment $\mathbf{s}^0$ is the cross product of any point $\mathbf{c}$ on the line times the direction $\mathbf{s}$. The symbol ε is a dual unit and its function is to separate the previous vectors; it has the property that $\varepsilon^2 = 0$. Because the specification of cross product, any points on the line can get same moment vector $\mathbf{s^0}$. Figure 4 shows the parameters that define a line.

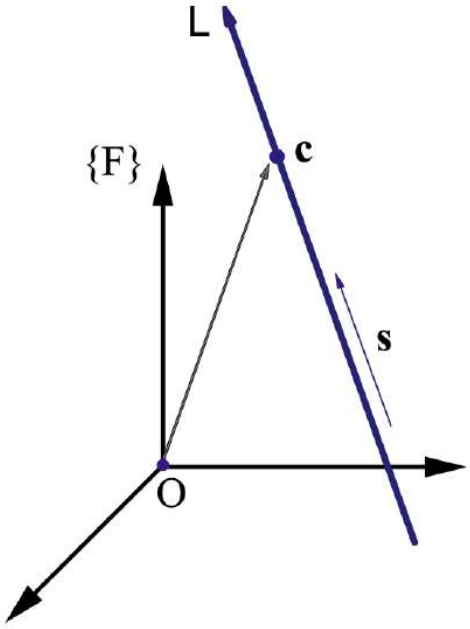


Figure 4. A line defined by using *Plücker coordinates*[3]

2.4 Forward Kinematics Equations

When we use a joint to connect two different parts, it can be though as a displacement, which is a rotation and/or translation about the joint axis, from one part to another. Since the end-effector connects to base through multiple joints, upon previous idea, we can compose the displacement on each joint and use the result to perform the displacement from the base to the end-effector.

2.4.1 Denavit-Hartenberg parameters

In order to create the local transformations base on the robotic chain, we identify the lines locating the joint axes S_i, and we compute the common normal lines $A_{i,i+1}$ between joint Si and

joint Si+1. We consider that the link joining two adjacent joints extends along the common normal line, regardless of its real shape. Local displacements will take us from local coordinate frame to local coordinate frame, from the base to the end-effector, see Figure 5.

For analyze the robotic system, we use a 4 by 4 homogeneous matrix to represent the local movement from joint to joint. The parameters in this matrix were generated by Denavit and

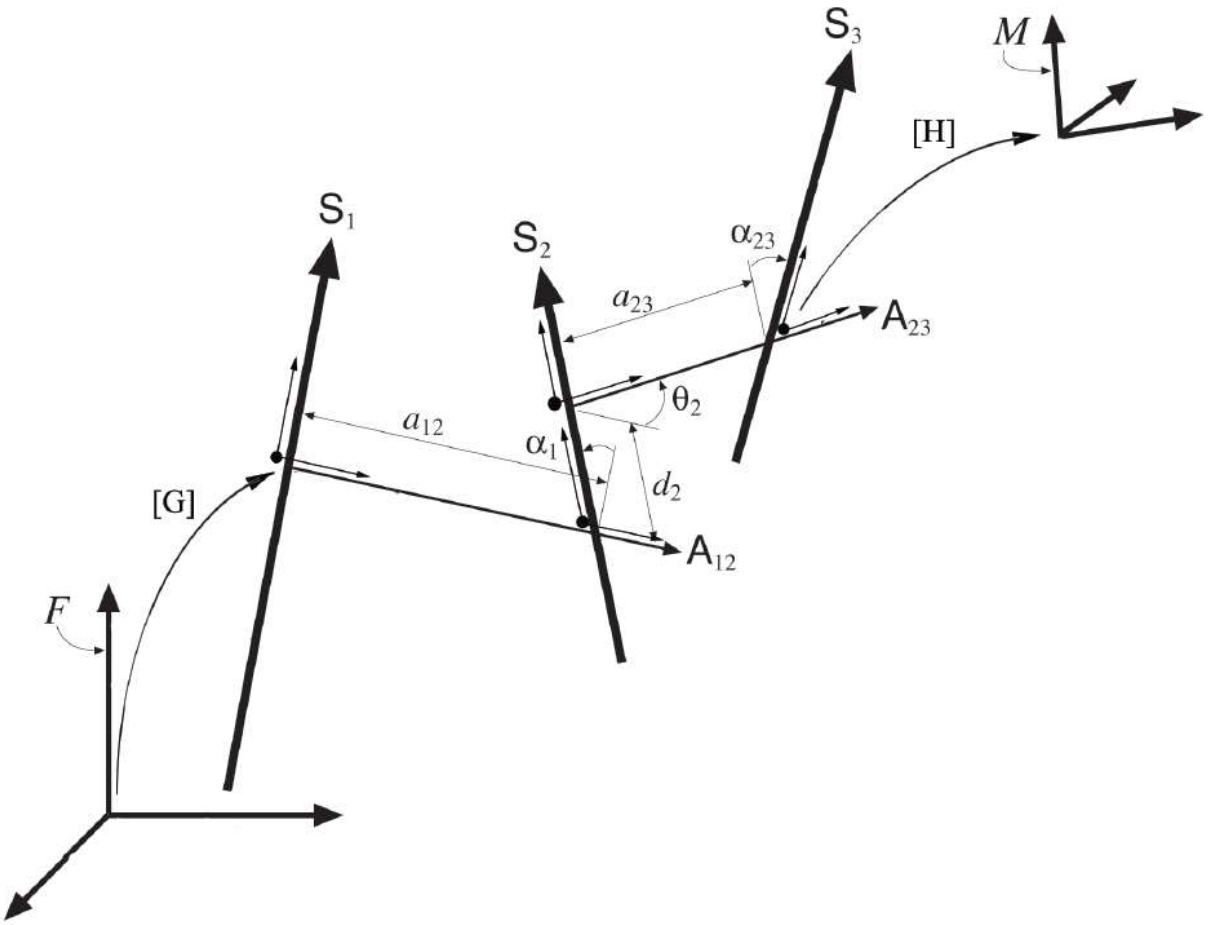


Figure 5. Local transformations along the links of a robot.[3]

Hartenberg in 1955, and now it has become a general tool in the robotic domain. They use only

four parameters to describe the local transformation, which was a screw displacement about Z or X axis. After years, there are some new versions of Denavit-Hartenberg parameters with small differences have been developed. Our notation is similar to Craig[3][5].

We locate the first basic frame with the Z axis along the first joint axis S_1 and the X axis along the common normal A_{12}; the origin of the frame is determined at the intersection of these two lines. Assume a displacement [*G*] that locates this first local frame with respect to the fixed frame, which is shown in Figure 5[3].

The X axis of the next local frame is same with before, but Z axis rotates to align with S_2, and its origin is located at intersection of these two lines. The third local frame has the same Z axis in the previous frame, and A_{23} is taken as the new X axis. Finally, define a displacement [*H*] from the last local frame, which is located at the last joint axis, to the moving frame attached to gripper.

Only four parameters are needed to transform from local frame to local frame: two correspond to the transformation along the link, which is an X-screw displacement, and two more to perform a Z-screw displacement about the joint. These parameters are:

- Twist angle $\alpha_{i-1,\,i}$: Angle between joint axes S_{i-1} and S_i measured about the common normal line $A_{i-1,\,i}$.
- Link length $a_{i-1,\,i}$: Distance between joint axes S_{i-1} and S_i measured along the common normal line $A_{i-1,i}$.

- Joint angle θ_i: Angle between previous common normal line $A_{i-1,\,i}$ and next common normal line $A_{i,\,i+1}$, measured about joint axes S_i. Notice that if joint S_i is a revolute joint, the joint angle will include the variable value of the joint rotation.
- Offset d_i: Distance between previous common normal line $A_{i-1,i}$ and next common normal line $A_{i,i+1}$, measured along joint axes Si. Notice that if joint Si is a prismatic joint, the offset will include the variable value of the slide.

Joint	$\alpha_{i-1,i}$	$a_{i-1,i}$	d_i	θ_i
1	0	0	d_1	θ_1
2	α_{12}	a_{12}	d_2	θ_2
3	α_{23}	a_{23}	d_3	θ_3

Table 1：Denavit-Hartenberg parameters[3]

The Denavit-Hartenberg parameters give us a simple way to analyze the robot geometry. These parameters usually put in a table like Table 1, which lists all Denavit-Hartenberg parameters in the robotic system is shown in Figure 5.

Equations 3 and 4 show the screw displacement based on Denavit-Hartenberg parameters,

$$[\boldsymbol{X}(\boldsymbol{\alpha_{i-1,i}}, \boldsymbol{a_{i-1,i}})] = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{a_{i-1,i}} \\ \mathbf{0} & \cos \boldsymbol{\alpha_{i-1,i}} & -\sin \boldsymbol{\alpha_{i-1,i}} & \mathbf{0} \\ \mathbf{0} & \sin \boldsymbol{\alpha_{i-1,i}} & \cos \boldsymbol{\alpha_{i-1,i}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{4}$$

$$[\boldsymbol{Z}(\boldsymbol{\theta_i}, \boldsymbol{d_i})] = \begin{bmatrix} \cos \boldsymbol{\theta_i} & -\sin \boldsymbol{\theta_i} & \mathbf{0} & \mathbf{0} \\ \sin_{\,\boldsymbol{i}} & \cos \boldsymbol{\theta_i} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{d_i} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{5}$$

2.4.2 Kinematic Equations

The local screw displacements created using the D-H parameters, together with the initial and final displacements [G] and [H], are multiplied together to obtain the displacement from the fixed frame to the frame attached to the end-effector,

$$[\boldsymbol{D}] = [\boldsymbol{G}]\{\prod_{\boldsymbol{i=1}}^{\boldsymbol{n}}[\boldsymbol{X}(\boldsymbol{\alpha_{i-1,i}}, \boldsymbol{a_{i-1,i}})][\boldsymbol{Z}(\boldsymbol{\theta_i}, \boldsymbol{d_i})]\}\,[\boldsymbol{H}]. \quad (6)$$

The 4 × 4 transform [D] is called the forward kinematics or simply the kinematics equations. It provides with all possible positions of the end-effector as a function of the joint variables. The kinematics equations solve the direct kinematics problem; for a given set of joint variables, the position of the end-effector is immediately calculated.

Chapter 3

Robotic Hand Design

From previous chapter we know that the number of joints will notably affect the structure of a robotic system, when all the possible end-effector is known. And if we want to build a robotic hand, we must know the number of different parts and the dimensions of each part. It hard to make the full of details of a robotic hand when we just analyze the structure. So we need an efficient way to describe the structure of a robotic system.

3.1 Topology of Robotic Hand

A tree topology for a kinematic chain has a set of common joints splitting on several chains, ending in multiple end-effectors[6][7]. Just as its name implies, the tree topology is a simplified tree shape, and every kinematic chain can be represented in this way. Figure 6 shows some topology graph of well-known hands.

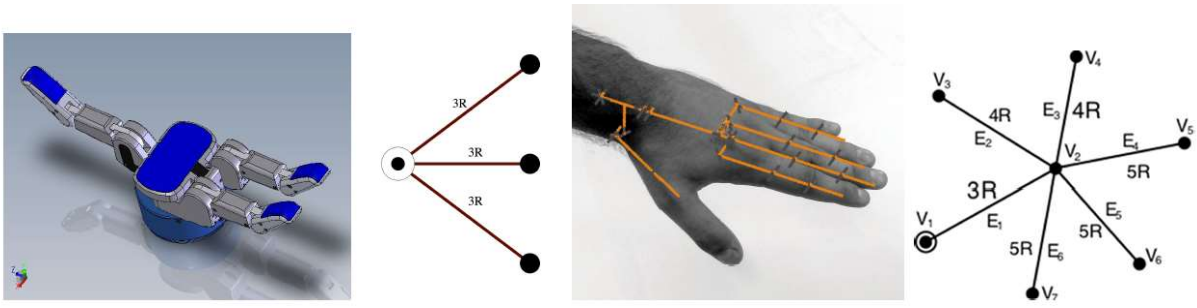


Figure 6: Tree topologies of BarrettHand and Human hand[6]

3.1.1 Compacted and Simplified Graphs

When formulating a synthesis problem, the internal loops in the hand structure can be substituted using a reduction process [6][8], which means that, if the tree topology of the hand has a closed loop, the reduction gets rid of the loops and use the most immediate path instead of those loops. Figure 7 represents how reduction process makes a tree topology of a hand simplified.

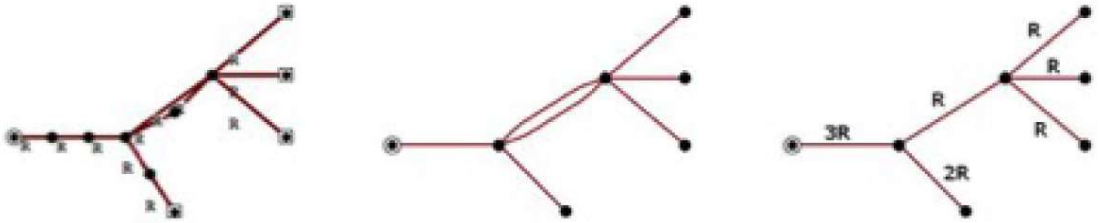


Figure 7: Original graph, left; compact graph, center; reduced graph, right.[6]

3.1.2 Tree Topology

Tree topologies are denoted as W – (B1 , B2 ,…, Bb), where W are the common joints and the " - " means a branch or splitting stage, with the branches contained in the parenthesis, every branch B_i defined by its type and number of joints. If we assume a hand and its joints is all revolute joint, the joint name will be hide and only number of joints is denoted. Figure 8 shows a example hand of one-jointed hand family, which is indicated as *R – (R, R, R, R, R)*, or 1 – (1, 1, 1, 1, 1) chain. The root vertex is indicated with a double circle.

We also add two arrays to represent the hand topology. We assume a numbering of the graph edges and define a parent-pointer array and a joint array. The length of both arrays is equal to number of edges of the graph after the reduction process is applied.

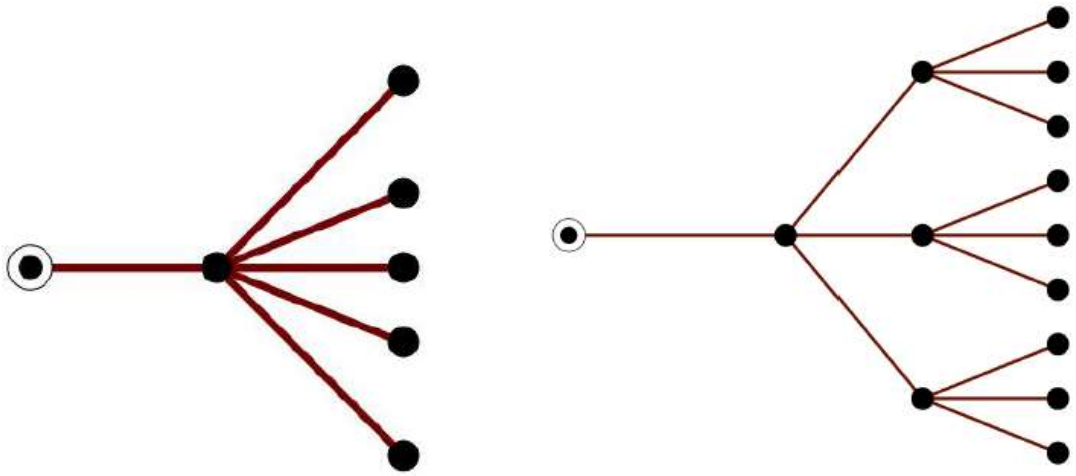


Figure 8: Tree graph for the 1 - (1, 1, 1, 1, 1) robotic hand with one palm, left; fractal hand with two splitting stages and three branches per split, right.[6]

The parent-pointer array implements the parent-pointer representation, where each element takes the value of the previous edge, the first edge being the one incident at the root vertex. Each element of the joint array contains the number and type of joints for each edge. As an example, for the tree topology shown in Figure 9, the parent-pointer array and joint array are defined as p = {0, 1, 1, 1, 3, 3, 3} and j = {2, 2, 1, 2, 3, 3, 3}.[6]

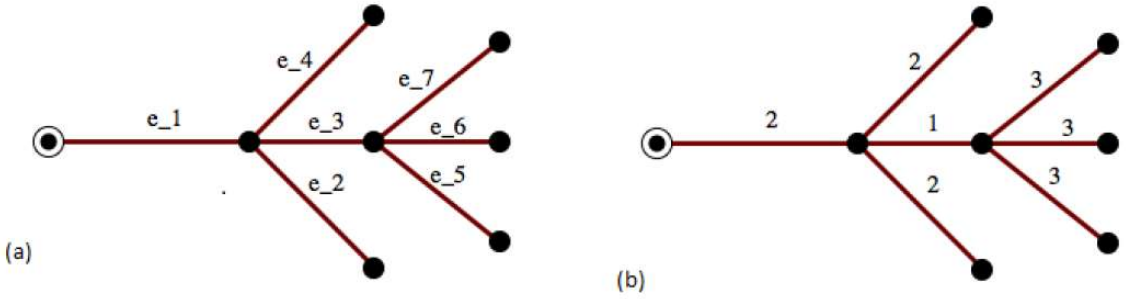


Figure 9: A five-fingered, two-palm hand topology. (a) indicates the numbering of the edges and (b) indicates the number of joints for each edge.[6]

3.2 Structural Synthesis

We always face a large number of topologies available for multi-fingered hands, so that it is necessary to find some criterion to select the most suitable topologies for a given task. While matching a workspace shape to a topology is still far from being accomplished except for simple cases[9], some boundary conditions on the candidate topologies can be based on the number of positions and the number of fingertips. So we select solvability as a criteria, which is describing the ability of the topology for being synthesized and reaching the workspace defined by the positions and without over-constraining any of its parts.

Algorithms have been developed [10] to find solvable topologies for a defined task. The task is a set of rigid motions, which is assumed to be set in the *SE*(3) group. It aims at finding all topologies that can satisfy the task for dimensional synthesis, given a set of user-defined restrictions. Those extra conditions bound a problem that otherwise may not be bounded, depending on the topology [6][11].

User-defined inputs includes the number of positions of the task *m*, the number of end-effectors *b* and the total number of edges of the graph *e*. The output is all possible topologies that suit the solvability criteria under the user-defined conditions. The formula in Eq. (7) is applied to do the enumeration,

$$\sum_{i=1}^{e} \boldsymbol{j_i} = \frac{(\boldsymbol{m-1})*\boldsymbol{6}*\boldsymbol{b}}{(\boldsymbol{m+3})} \quad (7)$$

where $\sum_{i=1}^{e} ji$ denote the total amount of joints and j_i are identify joints on the the *i*-th edge or serial chain. On each edge, the total number of joints has to be under 5 for synthesis purposes, as a serial chain of length 6 or higher does not impose any restriction on the motion.

The presented algorithm includes three steps:[6]

1. Find all possible joint arrays with length equal to e and satisfying Eq. (4), with entries between 1 and 5. The order of this part is $O(\frac{e*10^{e-1}}{2})$.
2. Find all possible tree structures (parent-pointer arrays *p*) which meet the input criteria, including the number of branches and number of edges.
3. Combine and check: For all possible parent pointer arrays, check all combinations of joint array and parent-pointer array for solvability. If the topology is solvable, then it becomes a candidate topology.

Table 2 shows the number of solvable topologies under given input that is the number of positons, number of end-effector and total number of edges. Since the number of solvable topologies is so big-the graphs for the topologies are not included.

Example	m	b	e	Solvable Topologies
Ex. 1	5	4	6	219
Ex, 2	21	5	8	34548

Table 2: Examples of type synthesis

3.3 Dimensional Synthesis

Dimensional synthesis seeks to find the position of the joint axes for a given topology, using as candidates some of the topologies which we obtained from structural synthesis. This synthesis sizes the hand in order to make every end-effector perform the displacements of the given task. In this section we present a summary of the design methodology; for details, see[8].

Assume a hand topology has b branches and a total of n_e joint axis S_i, and create the sets of ordered indices B_j of joints belonging to serial chain starting at the root and ending on end-effector j, for j =1,…,b. Given a simultaneous task for each fingertip, characterized by a set of m_p finite position $\hat{P}_{1k}^{b}$ and mv velocities $\dot{P}_{k}^{b}$, kinematic synthesis is applied by equating the forward kinematics equations of each branch to the relative displacement of the corresponding fingertip. Similarly, velocities can be defined for some of those task positions,

$$\mathbf{F}(\mathbf{S}, \Delta\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{cases} \widehat{\boldsymbol{P}}_{\boldsymbol{1k}}^{\boldsymbol{j}} - \prod_{\boldsymbol{i}\in\boldsymbol{B_j}} \boldsymbol{e}^{\frac{\Delta\boldsymbol{\theta}_{\boldsymbol{i}}^{\boldsymbol{k}}}{\boldsymbol{2}}}\mathbf{S}_{\boldsymbol{i}}, & \boldsymbol{k}=\boldsymbol{2},\dots,\boldsymbol{m_p} \\ & \boldsymbol{j}=\boldsymbol{1},\dots,\boldsymbol{b} \\ \dot{\boldsymbol{P}}_{\boldsymbol{k}}^{\boldsymbol{j}} - \sum_{\boldsymbol{i}\in\boldsymbol{B_j}} \dot{\boldsymbol{\theta}}_{\boldsymbol{i}}^{\boldsymbol{k}}\mathbf{S}_{\boldsymbol{j}}^{\boldsymbol{k}}, & \boldsymbol{k}=\boldsymbol{1},\dots,\boldsymbol{m_v} \\ & \boldsymbol{j}=\boldsymbol{1},\dots,\boldsymbol{b} \end{cases}, \tag{8}$$

where S_i^k is the i^{th} joint axes when it moved to position k. This includes a total 6(m - 1) b independent equations to be simultaneously solved. In most of the cases, this set of equations is solved using numerical methods to obtain a kinematic design [6].

3.3.1 Solvability of Tree Topologies for Exact Dimensional Synthesis

In previous part, we defined a solvable kinematic chain yields finite number of solutions, if we can get positive number result after we do dimensional synthesis to it.

When we dealing with tree topologies, the task sizing must be done while not over constraining any branch. This ensures that the equations of system can be solved simultaneously.

Let D_j^e be an $e \times 1$ vector containing the joint degrees-of-freedom for each edge of the contracted graph, and D_s^e be the $e \times 1$ vector containing the number of structural parameters (four per joint in the general case) for each edge of the contracted graph. Denote as D_{ee}^n the $b \times 1$ vector containing the degrees-of-freedom of the space of each end-effector, and D_c^n be the $b \times 1$ vector with the number of additionally imposed constraints (if any) for each branch. Define the vectors B_i as a $b_i \times 1$ vector of ones corresponding to the branches for subgraph i, and E_i as an $e_i \times 1$ vector of ones for the edges in the subgraph considered. Those can be easily computed using the end-effector path matrix and the incidence matrix of the graph. There are $2^b - 1$ possible subgraphs for any given rooted tree graph, including the overall graph. The maximum number of positions for the subgraph is given by

$$\boldsymbol{m_i} = \frac{\boldsymbol{D_s^e \cdot E_i - D_c^n \cdot B_i}}{\boldsymbol{D_{ee}^n \cdot B_i - D_j^e \cdot E_i}} + \mathbf{1}. \tag{9}$$

In addition to these, all different and non-isomorphic subgraphs that appear when exchanging the root node with each of the end-effectors need to be considered, see [11].

As a summary, an overall solution exists only when considering the solvability of all root-to-end-effector subgraphs, including root and end-effector switching. In this case, considering m_i as the number of positions for exact synthesis for a subgraph i with $i \in S$ the set

of all possible different end-effector subgraphs up to isomorphism, and m for the overall graph, the topology is solvable if

1. $m \in \mathbb{Q}^+$
2. $m \leq m_i, \forall m_i \in \mathbb{Q}^+, i \in \mathrm{S}$

3.3.2 Numerical Solution

Both the structural and the dimensional synthesis have been implemented in a solver. The structural synthesis follows a straightforward enumeration process and the dimensional synthesis is solved using a hybrid solver that returns a single solution for each run. This software is freely available at the project webpage [12].

Chapter 4

Robotic hand parts and database

In this chapter, the database structure and contents is described. This database can be used within an automatic design process to build model based on end-effector data and tree topology. All parts are built based on a commercial solid modeling software, SolidWorks Premium 2014 x64 Edition, which is one of the most commonly used packages in Mechanical design. This is combined with Office Excel as a tool to pass joint parameters in order to build the database.

One of the characteristics of the database is that it allows the user to customize the model. Parts changed with dimension changes, so the system provides with a few standard dimensions and the user also can type their input and modify the part dimensions to fit the requirements.

4.1. Structure of database

We define all parts as either end-effector or connector, considering the base part as a fixed end. In the connector category, we have multilink parts and single-link parts, the main difference between them is that multilink can link more than two joints. So we also call multilink to the palm parts, to relate them to the human hand.

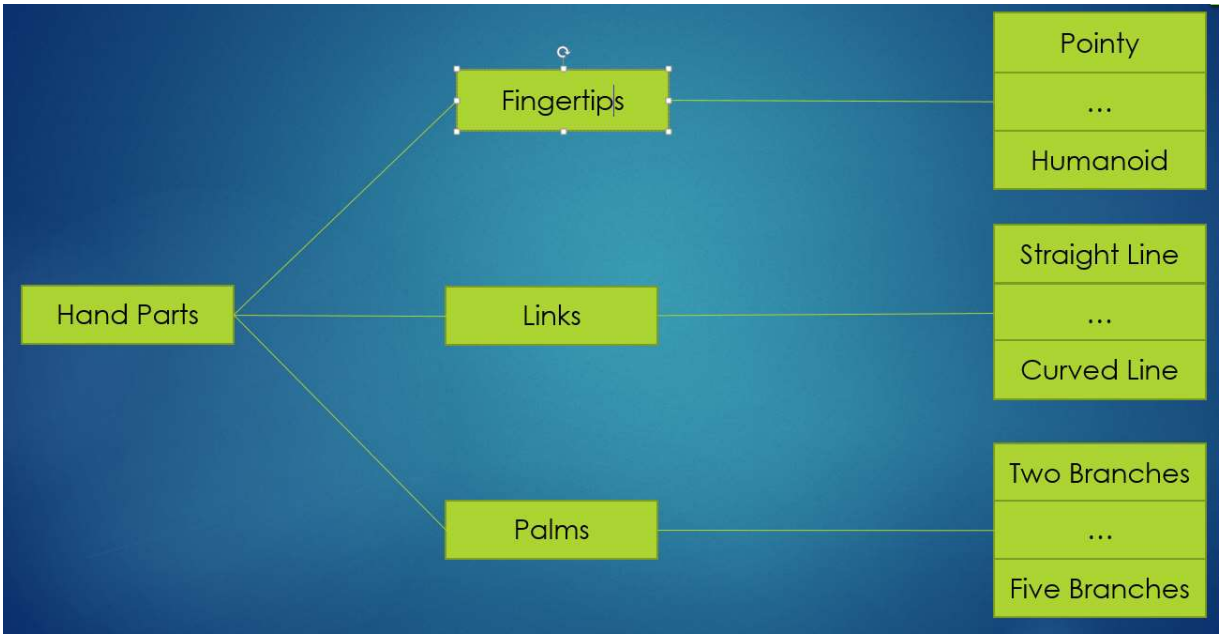


Figure 10: The Structure of Database

Like any other database, we have a structure for our database. Figure 10 shows the main parts included in our database:

- Fingertip: The main functional part, it will directly contact with the object or will have a free-movement task. The library presents three options, according to the functionality of the fingertip:
  1) Humanoid fingertip: have similar shape and function as human fingertip, there is one side with softer surface as main work surface;
  2) Pointy fingertip: a very sharp shape finger, the primary work part is top part. It has two kinds: frictional and frictionless. Since this kind fingertip is a pointy-contactor, it is easily to get a good simulation result.

3) Softer finger: As its name indicates, this kind of finger is wrapped by softer material for 360 degrees. Because its characteristics, this tip can working on several different direction.

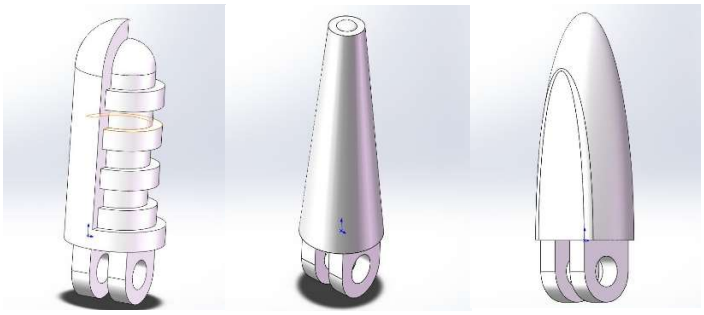


Figure 11: Softer Fingertip, left; Pointy Fingertip with friction, middle; Humanoid Fingertip, right

- Link: Compared to other two parts, this part may not be so special, the differences between the links currently present in the library are the shape of the cross-section and the curve. Curves can be straight, circle, ellipse even some unnamed line. And

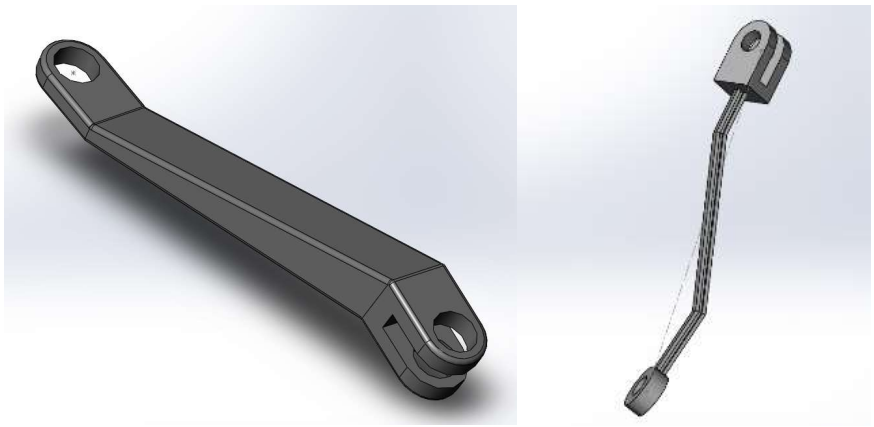


Figure 12: Sectional Straight Line link, left; Shaped Line link, right

square, circle, ellipse all can as a choice for cross-sectional shape, the main reason for making different lines, is to make any two links non-intrusive.

- Palm: If two or more link parts share same joint, then they become a palm. Normally, the number of branches is the typical difference among palm links, however the cross-sectional shape also yields different palm parts.

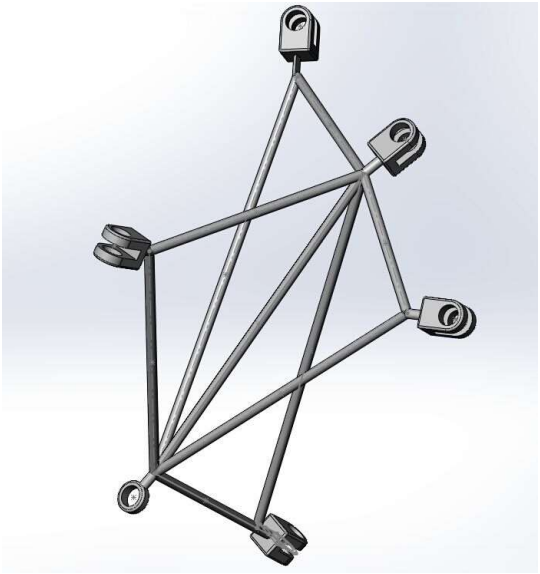


Figure 13: Five Branch Palm

4.2. Design Table

After several models for the links are designed, design tables are used to manage and edit our models. A design table allows you to build multiple configurations of parts or assemblies by specifying parameters in an embedded Microsoft Excel worksheet. The design table is saved in

the model document and is not linked to the original Excel file. Changes you make in the model are not reflected in the original Excel file. However, you can link the model document to the Excel file if you wish.

4.2.1. Set up the parameters of your parts

Before creating a design table, we need to denote all dimensions needed to make a particular part. Figure 14 shows a joint of link part, where the L, L1, L2 are defined as three different widths of the joint, and the right figure gives some parameters from top view of this

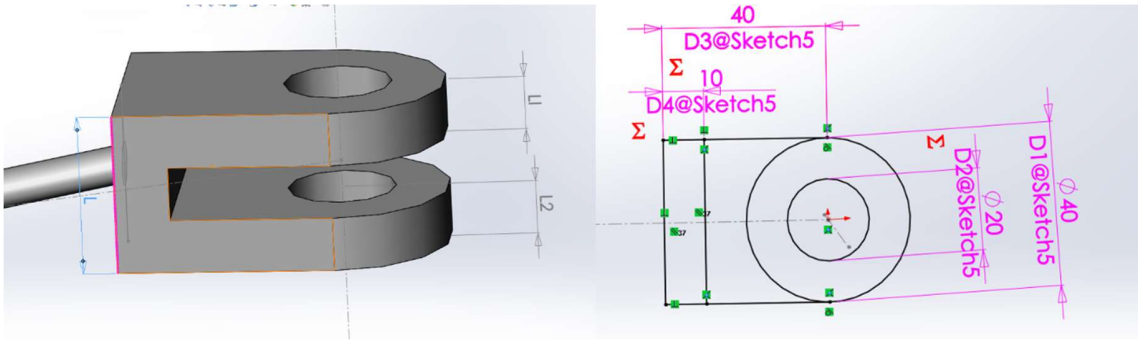


Figure 14: The parameter of a joint

joint. The labels give us two things: size and address. The above number is the basic size when you make this model, and the bottom part is the address of this dimension. This address is generated automatically when you build the model.

4.2.2. Create a design table

After dimensioning your model, click **Insert > Tables > Design Table** in an assembly document. There will be three choices for the design table sources:

- Blank: Inserts a blank design table where you fill in the parameters.

- Auto-create: Automatically creates a new design table, and loads all configured parameters and their associated values from a part or assembly.
- From file: References a Microsoft Excel table. Click Browse to locate the table. Then Links the Excel table to the model. When a design table is linked, any changes you make to the table outside of SolidWorks are reflected in the table within the SolidWorks model, and vice versa[13].

Figure 15 shows a simple example of a design table of a link part, which is generated by Auto create choice. The first row is the file name, the second is the address of dimension, and the last is the size of dimension.

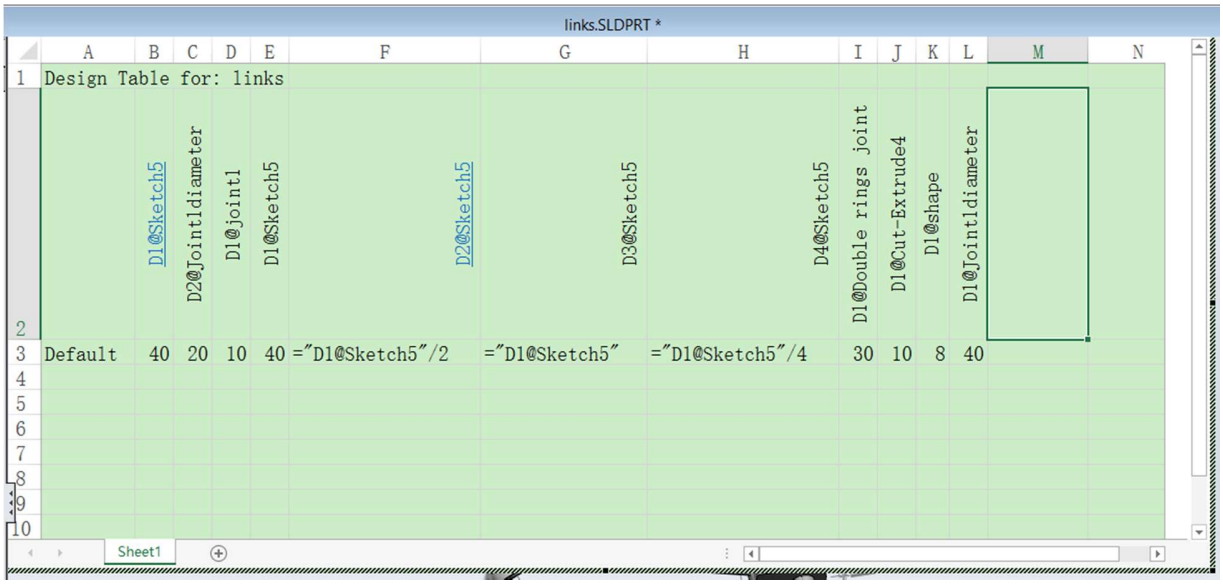
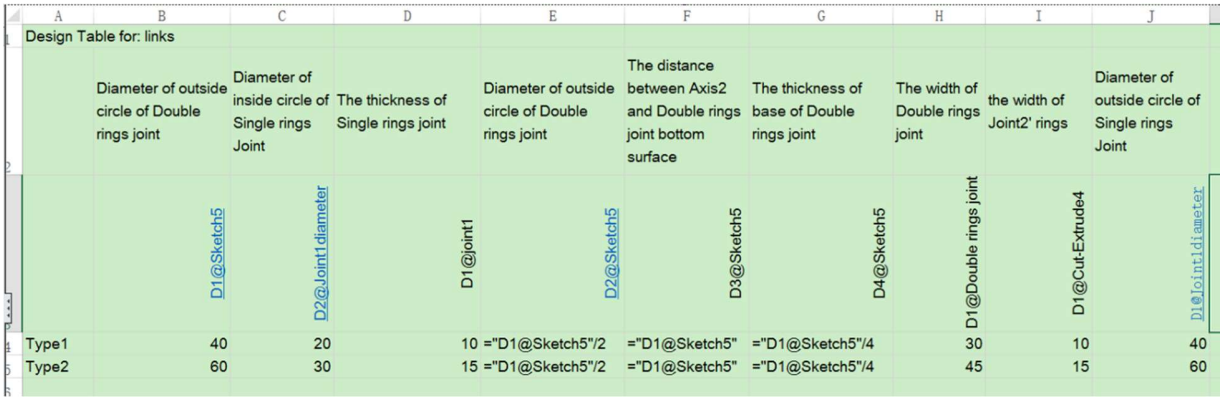


Figure 15: An example of design table

4.2.3. **Edit a design table**

Once the design table is created, we can make some details to make it easier to understand, we can add some notes or build connections between different parameters. Figure 16 follows from Figure 15, where first we can add any messages between first and second row. In



Design Table for: links

	Diameter of outside circle of Double rings joint	Diameter of inside circle of Single rings Joint	The thickness of Single rings joint	Diameter of outside circle of Double rings joint	The distance between Axis2 and Double rings joint bottom surface	The thickness of base of Double rings joint	The width of Double rings joint	the width of Joint2' rings	Diameter of outside circle of Single rings Joint
	D1@Sketch5	D2@Joint1diameter	D1@joint1	D2@Sketch5	D3@Sketch5	D4@Sketch5	D1@Double rings joint	D1@Cut-Extrude4	D1@Joint1diameter
Type1	40	20	10	="D1@Sketch5"/2	="D1@Sketch5"	="D1@Sketch5"/4	30	10	40
Type2	60	30	15	="D1@Sketch5"/2	="D1@Sketch5"	="D1@Sketch5"/4	45	15	60

Figure 16: A edited design table

this table, more imformation ccould be given to the parameters in second row and it will not give any additional effector to the model, unless it added any thing below address row. Then we build connection, it links coloum E, F and G to B. Since when we design a hand, there are some parametes whose dimension must be relative to other parameters for aesthetic, assembly and functional reasons. In this way ,we can just use a small amount of dimensions to assign all the dimension of a part.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Design Table for: Palm																					
2		D1@Sketch2	$STATE@circle0-1	$STATE@circle2-0	$STATE@circle2-1	$STATE@circle3-0	$STATE@circle3-1	$STATE@circle4-0	$STATE@circle4-1	$STATE@hex0-1	$STATE@hex2-0	$STATE@hex2-1	$STATE@hex3-0	$STATE@hex3-1	$STATE@hex4-0	$STATE@hex4-1	$STATE@Ring2	$STATE@Ring3	$STATE@Ring4	$STATE@Boss-Extrude3	$STATE@Boss-Extrude4	$STATE@Cut-Extrude1
3	CTwoBranch0.1	0.1	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
4	CTwoBranch0.2	0.2	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
5	CTwoBranch0.5	0.5	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
6	CTwoBranch1	1	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
7	CTwoBranch2	2	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
8	CTwoBranch5	5	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
9	CTwoBranch10	10	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
10	CTwoBranch20	20	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
11	CTwoBranch50	50	U	U	U	S	S	S	S	S	S	S	S	S	S	S	U	S	S	U	U	U
12	CThreeBranch0.1	0.1	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
13	CThreeBranch0.2	0.2	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
14	CThreeBranch0.5	0.5	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
15	CThreeBranch1	1	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
16	CThreeBranch2	2	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
17	CThreeBranch5	5	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
18	CThreeBranch10	10	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
19	CThreeBranch20	20	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U
20	CThreeBranch50	50	U	U	U	U	U	S	S	S	S	S	S	S	S	S	U	U	S	U	U	U

Sheet1

Figure 17: Change the state of some parts to a model by design table

Some parts include another type of field, like Figure 17, which is a design table for a palm. This field is denoted as $STATE@, followed by a feature name, as can be seen in the column header cell. It allows the user to change the state of the parts. When a palm is built, the library allows five branches at the maximum, and if the user does not want so many of them, he

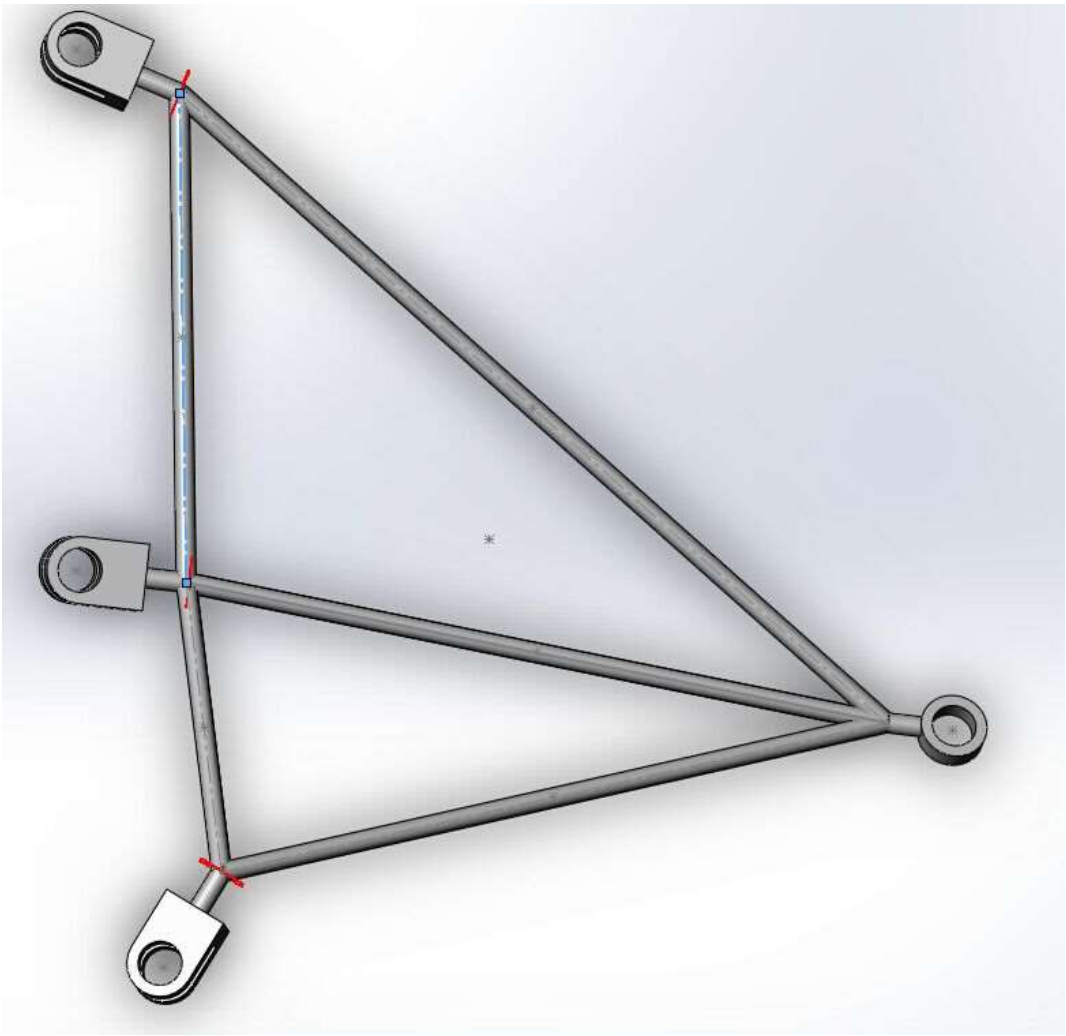


Figure 18: Three branch palm

can suppress any one of the branches. And Figure 17 also shows the options that we can fill below a state column.

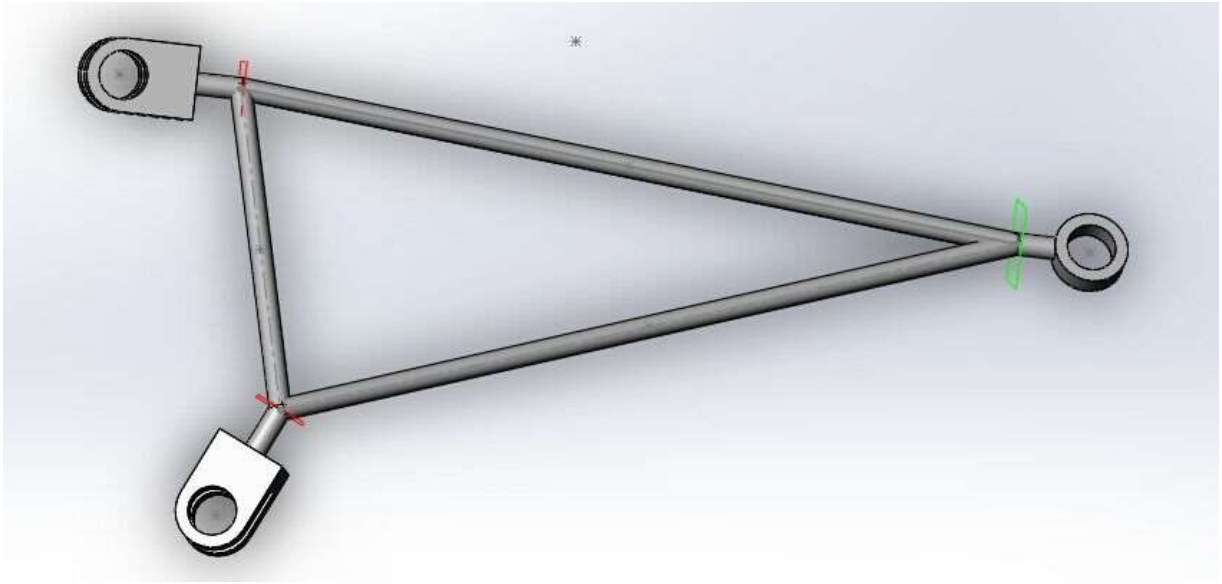


Figure 19: Two Branch Palm

Figure 18 and 19 show the model which include in Figure 17, they are row 19 and row 10. It is obvious that we just added a branch to turn Figure 19 into Figure 18, and this shows how the state in the design table affects the model. For any new model, just add the new number below every address column. For example in Figure 16, a type2's data after type 1 was added, and after closing the design table window, new member can be seen appearing under the configuration. In Figure 20, the type2 will automatically appear under type1. After finishing the data input, right click the Design Table Icon, which is also shown in Figure 20, and choose save

table to finish the first edit. In order to add more data in the future, just open configurations column, right click Design Table icon and choose “edit table”, and save the table after editing.

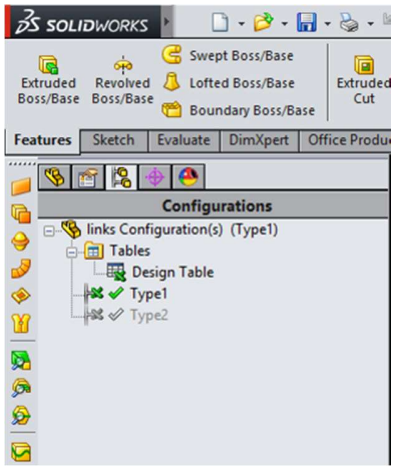


Figure 20: List of Model in Configurations

Chapter 5

Discussion of Results and Future Applications

Comparing with fully-implemented robotic hands, the parts in this database are not aiming to provide all the details. When we design a robotic hand, we may have two different aims. The first one is modeling the hand for- a motion task, like hand synthesis, another one is to build a hand where its shape is an important factor, like humanoid hands for prosthetics, which rely on mimicking and control to perform the task.

5.1. Hand Synthesis

From chapter 3, if we know the position of the end-effector task, the number of end-effectors and the number of branches, we can have finite results of hand topologies. And if we add other inputs which include specific joint positions, we will have few results about the dimension and structure.

But it is still difficult for the designer to be able to create a model with this basic information, as the designs greatly differ based on where the links are located along the joints for a given kinematic task. In addition to this, the designer must ensure the correct functioning of the hand by avoiding self-intersection of the parts, which is especially difficult for systems with several end-effectors such as robotic hands[14].

The current library and automatic process to generate a hand model has been proved useful to assess the complexity of the hand, the fulfillment of requirements such as overall size

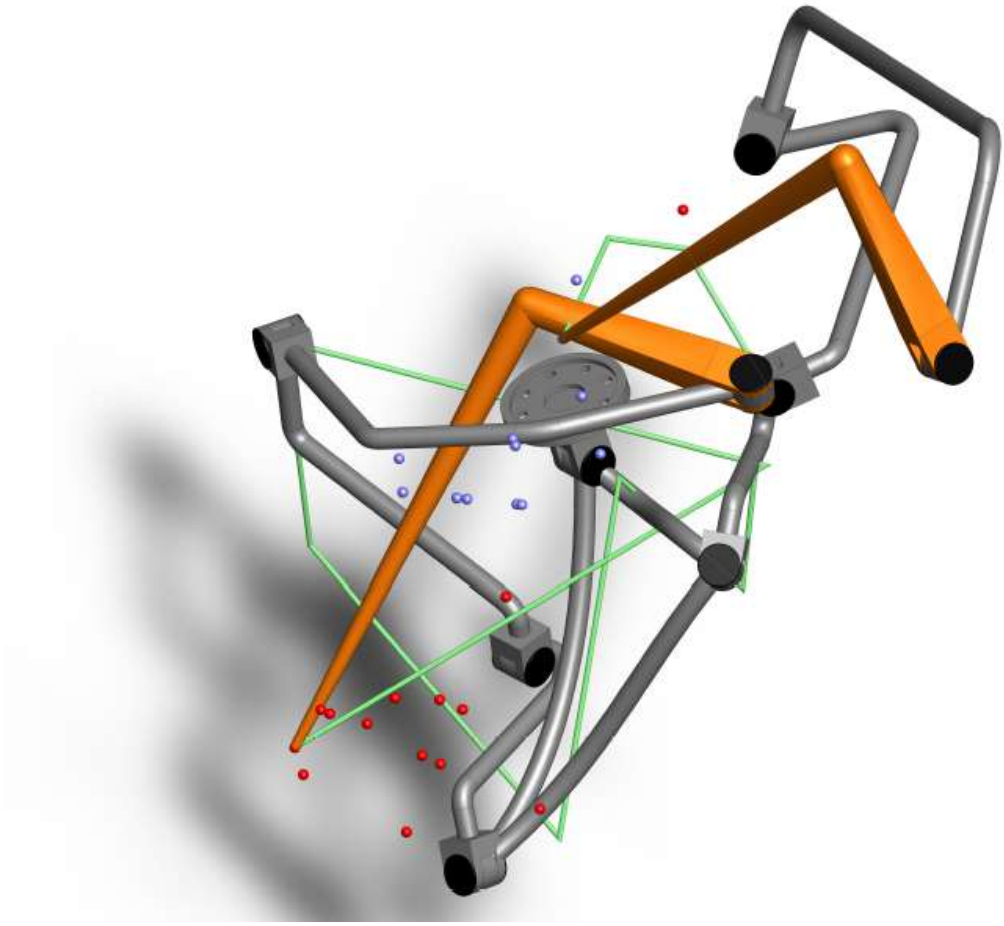


Figure 21: FINAL DESIGN FOR THE 1-(4, 4) HAND WITH ASK POSITIONS AND INITIAL COMMON-NORMAL LINES

or average dimensions, and its ability to perform the desired task. In this situation, my partner generate a program to finish all calculation part and build part. We have website can download this program freely.[14]

5.2. Library of parts as an assistant for the design

Human’s hands have many different sizes, this may affect its strength, but it does not affect its function. It means the hands structure determines function, dimensions affect the motion accuracy, and the operating system will tell us what the hands does. Under this idea, we can design a hand first, and then analyze it workspace by using the 3D-model or realizing the 3Dmodel using a 3D-printer.

References

[1] Kurt William Scott. *The Scott Hand – a decoupled solution to robotic prosthetics*. 2013.

[2] Barretthand web page. http://www.barrett.com/products-hand.htm.

[3] Alba Perez Gracia. *Kinematic of Robots.* 2013

[4] Releaux, F., 1875, *The Kinematics of Machinery: Outlines of a Theory of Machines*, Dover Publications, New York, translation of 1963.

[5] Craig, J. J., Introduction to Robotics, Mechanics and Control, Addison Wesley Publ. Co.,1989.

[6] Alba Perez Gracia. *Synthesis of Multi-Fingered Robotic Hands,* 2015.

[7] J. M. Selig. *Geometric Fundamentals of Robotics (Monographs in Computer Science).* SpringerVerlag, 2004.

[8] E. Simo-Serra and A. Perez Gracia. Kinematic synthesis using tree topologies. Mechanism and Machine Theory, 72 C:94-113, 2014.

[9] B. Batbold, Y. Yihun, J.S. Wolper, and A. Perez Gracia. *Exact workspace synthesis for rccr linkages*. In 2013 IFToMM Computational Kinematics Workshop, 2013.

[10] A. Tamimi and A. Perez Gracia. A robotic hand synthesis process for general tree topologies. In submitted to: Robotic Science and Systems, Rome, Italy, July 13-17, 2015.

[11] A. Makhal and A. Perez-Gracia. Solvable multi-fingered hands for exact kinematic synthesis. In *Advances in Robot Kinematics*, Ljubljiana, Slovenia, June 2014.

[12] Design of robotic hands webpage. http://robotics.engr.isu.edu/indexRH.html. Accessed: 2015-01-15.

[13] Solidworks help webpage. http://help.solidworks.c,om/2014/English/SolidWorks/sldworks/HIDD_DVE_DGNTBL.htm

[14] Neda Hassanzadeh, Xiangwei He, and Alba Perez-Gracia. *A design implementation process for robotic hand synthesis.* IDETC/CIE 2015, Boston, MA, August 2-5, 2015.